RTC Application Software

Software Classification Guide

Checkout and Launch Control System (CLCS)

84K01730-102

Approval:

Project Manager, CLCS (NASA)	Date	Project Manager, CLCS (USA)	Date

Concurrence:

NASA CLCS Application Software	Date	USA CLCS Application Software	Date
CLCS IV&V	Date		Date

NOTE: See "**Supporting Document Note**" on following page

PREPARED BY: Rich Ikerd, USK-284

Supporting Document Note: Acronyms and definitions of many common CLCS terms may be found in the following documents: CLCS Acronyms 84K00240 and CLCS Project Glossary 84K00250.

REVISION HISTORY

REV	DESCRIPTION	DATE

LIST OF EFFECTIVE PAGES				
Dates of issue of change pages are:				
Page No.	**A or D***	**Issue or Change No.**	**CR No.**	**Effective Date****

Table of Contents

RTC APPLICATION SOFTWARE

SOFTWARE CLASSIFICATION GUIDE

CHECKOUT AND LAUNCH CONTROL SYSTEM (CLCS)

1. INTRODUCTION

The Real Time Control (RTC) Application Software Classification Guide provides the methodology for classifying all software developed by, acquired by or sustained by the RTC Application Software organization.

1.1 PURPOSE

This practice establishes the methodology for classifying RTC Application Software. The classification of software ensures a standard, efficient and cost-effective approach to the development for all software having the same complexity and criticality.

1.2 SCOPE

This practice applies to all software developed by or acquired by the RTC Application Software organization.

1.3 AUTHORITY

This document is controlled by the Checkout and Launch Control System (CLCS) Application Software Chief or the appointed representative.

1.4 REFERENCE DOCUMENTS

The following documents were used in the development of or are referenced in this practice:

84K00054-200	RTC Application Software Development Plan
KMI 2410.6	KSC Software Management, Assurance and Engineering Policy

2. CLASSIFICATION PROCESS

All software development requires engineering, management and assurance activities of varying degrees to ensure accurate software is in compliance with requirements. The software classification process utilizes criticality and risk factors to assist the developing community in determining the minimum engineering, management and assurance activity required for each software element.

The factors to be used in the classification of software are performance, safety, cost, size, complexity, technology, testability and stability. A process has been created that gathers the pertinent information, organizes it and provides a classification matrix.

The classification process includes the breakdown of the software project into well defined classifiable software elements. These elements generally correspond to the Computer Software Components (CSC) or Computer Software Units (CSU) levels. This allows the classification at a granularity that can clearly identify the engineering, management and assurance activities that must be performed for the element. In the interest of economy and efficiency, each element is classified independently of all others.

2.1 PROCESS DEFINITION

The process of software classification is relatively straightforward given the guidelines of this practice. There is some subjectivity in the determination process, so it is incumbent upon the developers to consult with their users and their peers in an attempt to arrive at the most appropriate classification possible. Since this classification is used to determine the level of oversight necessary in the development/validation process, it is imperative that sound engineering principles be applied during the classification process.

The following process shall be used to classify a software component:

1. Determine the appropriate criticality level based on the guidelines defined in Section 2.2 Criticality Levels.

2. Determine the appropriate complexity level based on the guidelines defined in Section 2.3 Risk Levels.

3. Using the matrix defined in Section 2.4 Software Classification Determination, assign the appropriate software classification to the component.

4. The Overview Design Specification section for the component shall be updated to indicate the classification, the criticality level and the complexity level. This classification will be reviewed by the APTeam during the Requirements Review Panel to ensure consistency across CSCIs.

2.2 CRITICALITY LEVELS

Criticality levels are assigned based on the consequences of a system failure resulting from a software error. An impact value is assigned to each of the three criticality components (performance and operation, safety and development cost/schedule) based on the criteria in Table 2-1. The average of these three values is the assigned criticality.

The evaluation of criticality must also consider mitigating factors to the severity of consequences resulting from a function's failure. Redundancy of a function will mitigate the loss of a single string implementing that function. There are process and procedural controls that are employed as an additional level of criticality mitigation. The physical hardware design may include built-in protections (e.g., relief valves, circuit breakers) that provide mitigation should software function improperly. These mitigating factors are distinct from the Risk level (see next section). It is possible to have low criticality, but high risk.

Table 2-1 Criticality Rating Criteria

CRITICALITY CATEGORIES, RATING CRITERIA AND NOTES				
Criticality Driver	Catastrophic Impact Value = 4	Critical Impact Value = 3	Moderate Impact Value = 2	Low Impact Value = 1
Performance and Operation	Failure of the system could cause loss of ability to launch vehicles for extended periods of time, or loss of capability to perform all mission objectives. Failure is not mitigated.	Failure could cause loss of a critical function that does not result in the inability to launch vehicles. Failure could also cause lengthy maintenance downtime, loss of ability to perform multiple mission objectives or major damage to a subsystem. Failure is partially mitigated.	Failure could cause loss of a single mission objective or reduction in operational capability. Failure is fully mitigated. A work-around exists.	Failure could cause inconvenience (e.g. re-run of programs, restart of computer, manual intervention).
	Note: Because of the migration to CLCS from the existing CCMS system, there are few application pieces that meet this category. GLS is rated a level 4.	Note: Major Prelaunch sequencers are in this category. Most other sequencers have a workaround (thus a lower rating)	Note: The bulk of applications fall into this category. There are workarounds, although they may not be the most efficient method of processing.	Note: This category is applicable to the lower level reusable components, which can be easily supported by manual operator override.
Safety	Failure could result in loss of life or vehicle or cause severe personal injury.	Failure could result in non-disabling personal injury, serious occupational illness, or loss of emergency procedures.	Failure could result in minor injury	No safety implications.
	Note: this covers High Power and Major Hazard systems.	Note: This is most control logic. Also High Power system(w/o associated major Hazards)	Note: Most systems fall into this category, because of the bulk of procedural (OMI) controls that are established to preclude injury.	Note: Most avionics systems fit this category, because they have little potential to cause any threat to safety.
Development Cost/Schedule	Failure could result in cost overruns large enough to result in inability to achieve operational capability.	Failure could result in large cost and schedule over-runs. Alternate means to implement function are not available.	Failure results in significant schedule delay. Alternate means to implement function are available but at reduced operational capability. Full operational capability delayed.	Failure results in minor impact to cost and schedule. Problems are easily corrected with insignificant impact to cost and schedule.
Note: Because CLCS is a transition from an existing functional system, categories 3 and 4 do not apply. CCMS is an alternate means to support the functionality.	Note: N/A	Note: N/A	Note: The bulk of initial development falls into this category (impact is a delay of full operational capability).	Note: As development matures, development impact decreases (problem space is well defined and is mature).

2.3 RISK LEVELS

Software component risk is based on the complexity, maturity of technology, requirements definition and stability and testability. A risk value is assigned to each of the four risk components based on the criteria in Table 2-2. The average of these values is the assigned criticality.

Risk is independent from Criticality and should be rated based on the criteria specified. The tendency is to rate high criticality items with a high risk (and this is not necessarily always true).

Table 2-2 Risk Assessment Criteria

RISK CATEGORIES, RATING CRITERIA AND NOTES			
Risk Drivers	High Risk Value = 3	Moderate Risk Value = 2	Low Risk Value = 1
Complexity	• Highly complex control logic operations • Unique devices/complex interfaces • Many interrelated components • Function uses different end items in different modes of system operation	• Moderately complex control logic operations • May be device dependent • Moderately complex interfaces • Several interrelated components • Function behaves differently in different modes of system operation	• Simple control/logic operations • Device independent • Function operates in only one mode of system operation
	Note: This covers most major Integrated (complex) sequencers	Note: Multi-mode system operations.	Note: Non-integrated software, standalone system tests, display S/W.
Maturity of Technology	• New/Unproven algorithms, languages and support environments • High probability for redesign • Little or not experience base in this application	• Proven on other systems with different applications • Moderate experience base	• Proven on other systems with same application • Mature experience • High reuse
	Note: Because of the build a little-test a little process, this applies only to the initial IPT (HMF)	Note: Most applications fit this criteria. Builds upon the HMF base.	Note: When an IPT moves to a second or third phase, the experience base is robust, this category is appropriate.
Requirements Definition & Stability	• Rapidly changing, baselines not established • Many organizations required to define requirements • Much integration required	• Potential for some changes • Some integration required	• Solid requirements - little potential for change • Little to no integration required
Note: Because there is an existing requirements base, there is little risk that requirements will change once they are baselined.	Note: For major integrated operations.	Note: Most systems fall in this category	Note: Applies when existing GOAL requirements are mature enough for a simple transformation to CLCS.
Testability	• Difficult to test • Requires the analysis of a large amount of data to determine acceptability of results • Many operational environment and input variations	• Requires some test data analysis to determine acceptability of results • Moderate amount of operational environment and input variations	• Acceptability of test results easily determined • Few operational environment and input variations
Note: With the hierarchical design methodology, even the most complex sequence is broken into smaller parts.	Note: Very complex sequences fit this category.	Note: The bulk of systems require some analysis and have various input conditions.	Note: Applies to the simple low level reuse items.

2.4 SOFTWARE CLASSIFICATION DETERMINATION

Software component classification is based a combination of the assigned criticality and risk levels. Defining the applications classification properly ensures the appropriate level of engineering, management and quality oversight is applied. There are three classification levels of in the RTC Application Software environment. A component "bubbles" to the highest level of criticality applicable. Table 2-3 provides a matrix for determining the classification based on the previously assigned criticality and risk levels. Even though safety is not specifically addressed, all components with a safety critically of 3 or 4 fall into the sensitive or critical categories. The matrix implicitly addresses safety concerns, when there is a safety critical component, more than the minimal test is required.

Table 2-3 Software Classification Matrix

	Risk		
Criticality	***3***	***2***	***Low***
4	Critical	Critical	Sensitive
3	Critical	Sensitive	Operational
2	Sensitive	Sensitive	Operational
1	Operational	Operational	Operational

The engineering, management and assurance oversight required for a component is based on its classification. The definition of this oversight is provided in the RTC Application Software Development Plan 84K00054-200.

2.4.1 Critical

Critical components are the "front line" items whose operation must be ensured to allow a successful commitment of the vehicle to space flight. The criticality of these components requires a level of quality support above the baseline. IV&V personnel are anticipated to participate at this level. Critical software includes some high energy control logic components, Launch Sequencing/Launch Commit and Cryo load sequences. It does not include most system test/monitor support software. Approximately 10% of all RTC software is classified critical.

2.4.2 Sensitive

Sensitive components encompass the control and monitor of items that, if not operated properly, could result in personnel injury or hardware damage. This class of items includes "reactive" responses to off-nominal conditions and the control of high energy systems (e.g., cryogenics, hydraulics, power units, etc.). The sensitivity of these components requires a minimum level of independent oversight to ensure proper validation of requirements. Product Assurance Engineer (PAE) support is required. Approximately 25% of the application space fits the sensitive category. IV&V personnel may audit this function to ensure procedure compliance.

2.4.3 Operational

Operational support components make up a large portion of the RTC Application Software elements and are classified as non-critical. These are used for "baby-sit" support and the control and monitoring of non-hazardous systems and the monitoring of systems where "reactive control" has been delegated to external systems (e.g., a smart HIM, on-board sequence, etc.). An operational support component would not have an associated time critical safing function. The benign nature of these components requires less external oversight. IPTs are chartered to validate these components. PAE auditing is used to ensure procedural compliance. The operations components are essentially tested by a Responsible Organization Representative (ROR) without any explicit oversight. Approximately 65% of the applications space fits the operational category.